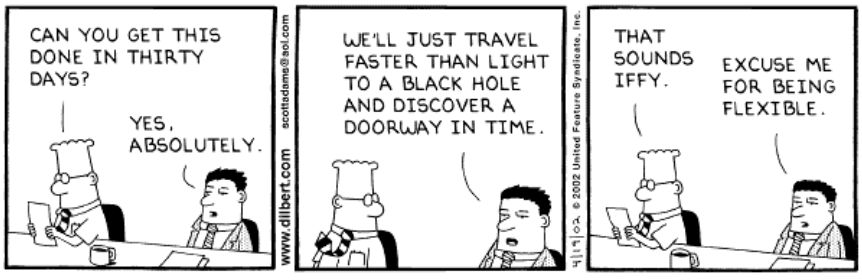
MA/CSSE 474
Theory of Computation

Nondeterminism

Parsing (top-down, bottom-up)

Your Questions?

- Previous class days' material
- Reading Assignments
- HW11 or 12 problems
- Anything else

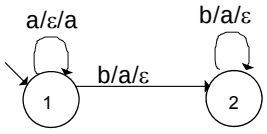


Copyright © 2002 United Feature Syndicate, Inc.

More on Nondeterminism
Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

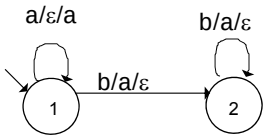
Start with the case where $n = m$:



More on Nondeterminism
Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

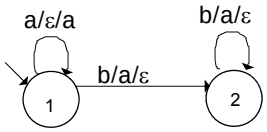
Start with the case where $n = m$:



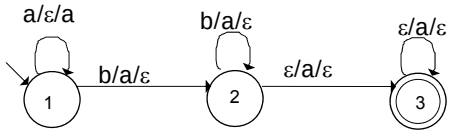
- If stack and input are empty, halt and reject.
- If input is empty but stack is not ($m > n$) (accept):
- If stack is empty but input is not ($m < n$) (accept):

More on Nondeterminism
Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

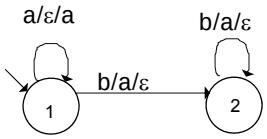


- If input is empty but stack is not ($m > n$) (accept):

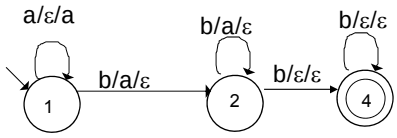


More on Nondeterminism
Accepting Mismatches

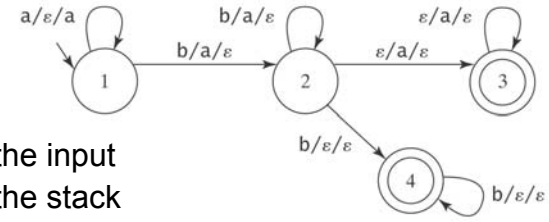
$L = \{a^m b^n : m \neq n; m, n > 0\}$



- If stack is empty but input is not ($m < n$) (accept):



$L = \{a^m b^n : m \neq n; m, n > 0\}$



- State 4: Clear the input
- State 3: Clear the stack
- A non-deterministic machine!

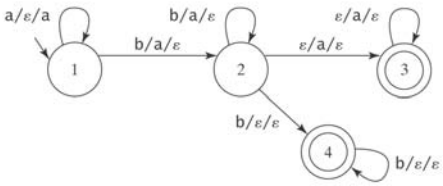
  What if we could

  detect end of input (as we can in real-world situations)?

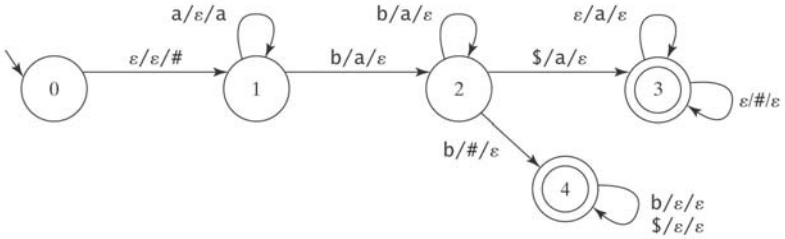
  detect empty stack?
- Add end-of-input marker $ to Σ
- Add bottom-of-stack marker # to Γ

Reducing Nondeterminism

- Original non-deterministic model



- With the markers:



The Power of Nondeterminism

Consider $A^nB^nC^n = \{a^nb^nc^n: n \geq 0\}$.

PDA for it?

The Power of Nondeterminism

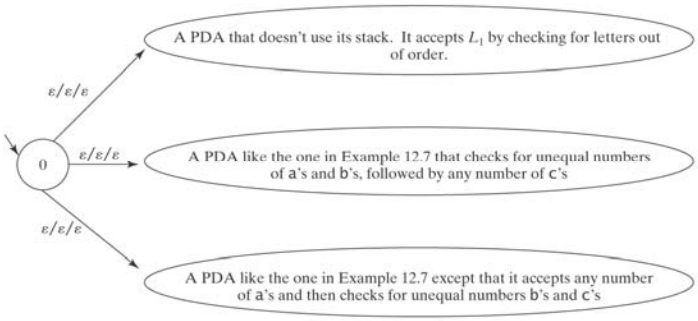
Consider $A^nB^nC^n = \{a^nb^nc^n: n \geq 0\}$. PDA for it?

Now consider $L = \neg A^nB^nC^n$. L is the union of two languages:

1. $\{w \in \{a, b, c\}^* :$ the letters are out of order$\}$, and

2. $\{a^ib^jc^k: i, j, k \geq 0$ and $(i \neq j$ or $j \neq k)\}$ (in other words, unequal numbers of a's, b's, and c's).

A PDA for $L = \neg A^nB^nC^n$



Are the Context-Free Languages Closed Under Complement?

$\neg A^nB^nC^n$ is context free.

If the CF languages were closed under complement, then

$$\neg\neg A^nB^nC^n = A^nB^nC^n$$

would also be context-free.

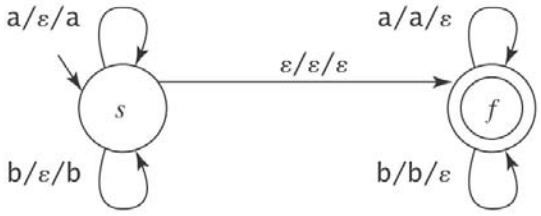
But we will prove that it is not.

$L = \{a^n b^m c^p : n, m, p \geq 0 \text{ and } n \neq m \text{ or } m \neq p\}$

$S \rightarrow NC$	/* $n \neq m$, then arbitrary c's
$S \rightarrow QP$	/* arbitrary a's, then $p \neq m$
$N \rightarrow A$	/* more a's than b's
$N \rightarrow B$	/* more b's than a's
$A \rightarrow a$	
$A \rightarrow aA$	
$A \rightarrow aAb$	
$B \rightarrow b$	
$B \rightarrow Bb$	
$B \rightarrow aBb$	
$C \rightarrow \varepsilon \mid cC$	/* add any number of c's
$P \rightarrow B'$	/* more b's than c's
$P \rightarrow C'$	/* more c's than b's
$B' \rightarrow b$	
$B' \rightarrow bB'$	
$B' \rightarrow bB'c$	
$C' \rightarrow c \mid C'c$	
$C' \rightarrow C'c$	
$C' \rightarrow bC'c$	
$Q \rightarrow \varepsilon \mid aQ$	/* prefix with any number of a's

More on PDAs

A PDA for $\{ww^R : w \in \{a, b\}^*\}$:



What about a PDA to accept $\{ww : w \in \{a, b\}^*\}$?

PDAs and Context-Free Grammars

Theorem: The class of languages accepted by PDAs is exactly the class of context-free languages.

Recall: context-free languages are languages that can be defined with context-free grammars.

Restate theorem:

Can describe with context-free grammar

=

Can accept by PDA

Going One Way

***Lemma**:* Every context-free language is accepted by some PDA.

Proof (by construction):

The idea: Let the stack do most of the work.

Two approaches:

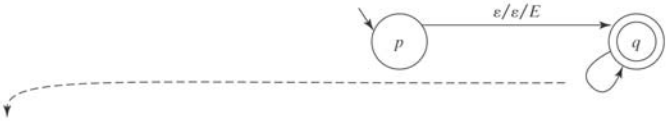
- Top down
- Bottom up

Top Down

The idea: Let the stack keep track of expectations.

Example: Arithmetic expressions

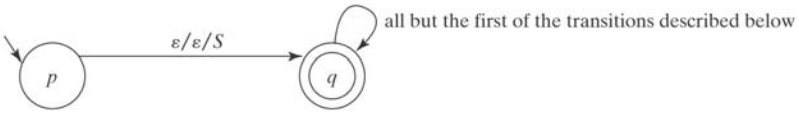
$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$



(1) $(q, \varepsilon, E), (q, E+T)$
(2) $(q, \varepsilon, E), (q, T)$
(3) $(q, \varepsilon, T), (q, T*F)$
(4) $(q, \varepsilon, T), (q, F)$
(5) $(q, \varepsilon, F), (q, (E))$
(6) $(q, \varepsilon, F), (q, id)$
(7) $(q, id, id), (q, \varepsilon)$
(8) $(q, (, (\ )), (q, \varepsilon)$
(9) $(q,),)\ )), (q, \varepsilon)$
(10) $(q, +, +), (q, \varepsilon)$
(11) $(q, *, *), (q, \varepsilon)$

A Top-Down Parser

The outline of *M* is:



$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

- The start-up transition $((p, \varepsilon, \varepsilon), (q, S))$.
- For each rule $X \rightarrow s_1 s_2 \ldots s_n$. in R, the transition: $((q, \varepsilon, X), (q, s_1 s_2 \ldots s_n))$.
- For each character $c \in \Sigma$, the transition: $((q, c, c), (q, \varepsilon))$.

Another Example

$L = \{a^n b^m c^p d^q : m + n = p + q\}$

(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$

input = a a b c c d

0 $(p, \varepsilon, \varepsilon), (q, S)$
1 $(q, \varepsilon, S), (q, aSd)$
2 $(q, \varepsilon, S), (q, T)$
3 $(q, \varepsilon, S), (q, U)$
4 $(q, \varepsilon, T), (q, aTc)$
5 $(q, \varepsilon, T), (q, V)$
6 $(q, \varepsilon, U), (q, bUd)$
7 $(q, \varepsilon, U), (q, V)$
8 $(q, \varepsilon, V), (q, bVc)$
9 $(q, \varepsilon, V), (q, \varepsilon)$
10 $(q, a, a), (q, \varepsilon)$
11 $(q, b, b), (q, \varepsilon)$
12 $(q, c, c), (q, \varepsilon)$
13 $(q, d, d), (q, \varepsilon)$

trans ***state*** ***unread input*** ***stack***

Notice the Nondeterminism

Machines constructed with the algorithm are often nondeterministic, even when they needn't be. This happens even with trivial languages.

Example: $A^nB^n = \{a^n b^n : n \geq 0\}$

A grammar for A^nB^n is:

[1] $S \rightarrow aSb$
[2] $S \rightarrow \varepsilon$

A PDA M for A^nB^n is:

(0) $((p, \varepsilon, \varepsilon), (q, S))$
(1) $((q, \varepsilon, S), (q, aSb))$
(2) $((q, \varepsilon, S), (q, \varepsilon))$
(3) $((q, a, a), (q, \varepsilon))$
(4) $((q, b, b), (q, \varepsilon))$

But transitions 1 and 2 make M nondeterministic.

A directly constructed machine for A^nB^n can be deterministic.

Constructing deterministic top-down parsers major topic in CSSE 404.

The Other Way to Build a PDA - Directly

$L = \{a^n b^m c^p d^q : m + n = p + q\}$

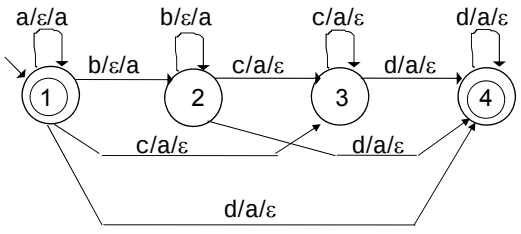
(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$

input = a a b c d d

The Other Way to Build a PDA - Directly

$L = \{a^n b^m c^p d^q : m + n = p + q\}$

(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$



input = a a b c d d

Example for practice later

$L = \{a^n b^* a^n\}$

(1) S → ε *
(2) S → B
(3) S → aSa
(4) B → ε
(5) B → bB

input = a a b b a a

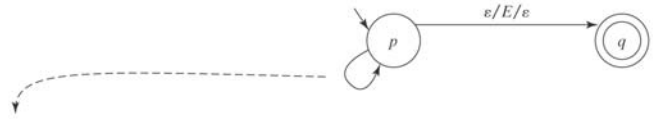
0 (p, ε, ε), (q, S)
1 (q, ε, S), (q, ε)
2 (q, ε, S), (q, B)
3 (q, ε, S), (q, aSa)
4 (q, ε, B), (q, ε)
5 (q, ε, B), (q, bB)
6 (q, a, a), (q, ε)
7 (q, b, b), (q, ε)

trans	*state*	*unread input*	*stack*
	p	a a b b a a	ε
0	q	a a b b a a	S
3	q	a a b b a a	aSa
6	q	a b b a a	Sa
3	q	a b b a a	aSaa
6	q	b b a a	Saa
2	q	b b a a	Baa
5	q	b b a a	bBaa
7	q	b a a	Baa
5	q	b a a	bBaa
7	q	a a	Baa
4	q	a a	aa
6	q	a	a
6	q	ε	ε

Bottom-Up Parser

The idea: Let the stack keep track of what has been found.

(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow (E)$
(6) $F \rightarrow id$



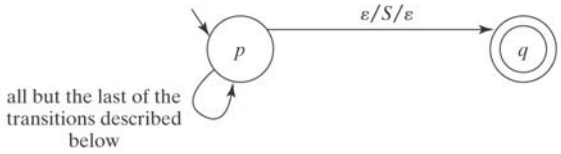
Reduce Transitions:
(1) (*p*, ε, *T* + *E*), (*p*, *E*)
(2) (*p*, ε, *T*), (*p*, *E*)
(3) (*p*, ε, *F* ∗ *T*), (*p*, *T*)
(4) (*p*, ε, *F*), (*p*, *T*)
(5) (*p*, ε,)*E*( ), (*p*, *F*)
(6) (*p*, ε, id), (*p*, *F*)

Shift Transitions
(7) (*p*, id, ε), (*p*, id)
(8) (*p*, (, ε), (*p*, ()
(9) (*p*,), ε), (*p*,))
(10) (*p*, +, ε), (*p*, +)
(11) (*p*, ∗, ε), (*p*, ∗)

Example: id + id * id. **Do the derivation first**

A Bottom-Up Parser

The outline of M is:



$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

- The shift transitions: $((p, c, \varepsilon), (p, c))$, for each $c \in \Sigma$.
- The reduce transitions: $((p, \varepsilon, (s_1 s_2 \ldots s_n.)^R), (p, X))$, for each rule $X \rightarrow s_1 s_2 \ldots s_n$. in G. **Un does an application of this rule.**
- The finish up transition: $((p, \varepsilon, S), (q, \varepsilon))$.

A top-down parser discovers a leftmost derivation of the input string (If any).
Bottom-up discovers rightmost derivation (in reverse order)